



Last Line of Defense

Florian Brand
Fedora Ambassador



*The firewall is open, the only
admin is on vacation and
Bugtrack reports an exploit exactly
for the version of Apache used..*

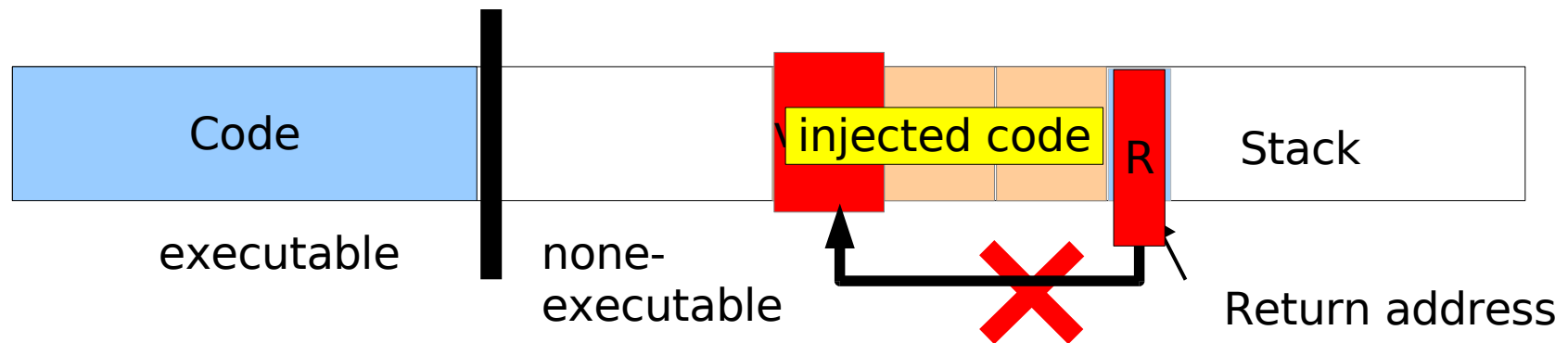




Agenda

- Attack Methods
- Defenses
 - Exec Shield
 - Compiler/Glibc Features
- SELinux
 - Basics
 - Example: Apache
 - New Features

Attack Method: Buffer Overflow



Exec Shield

- Prevents the execution of writable memory
- Makes it difficult to predict addresses

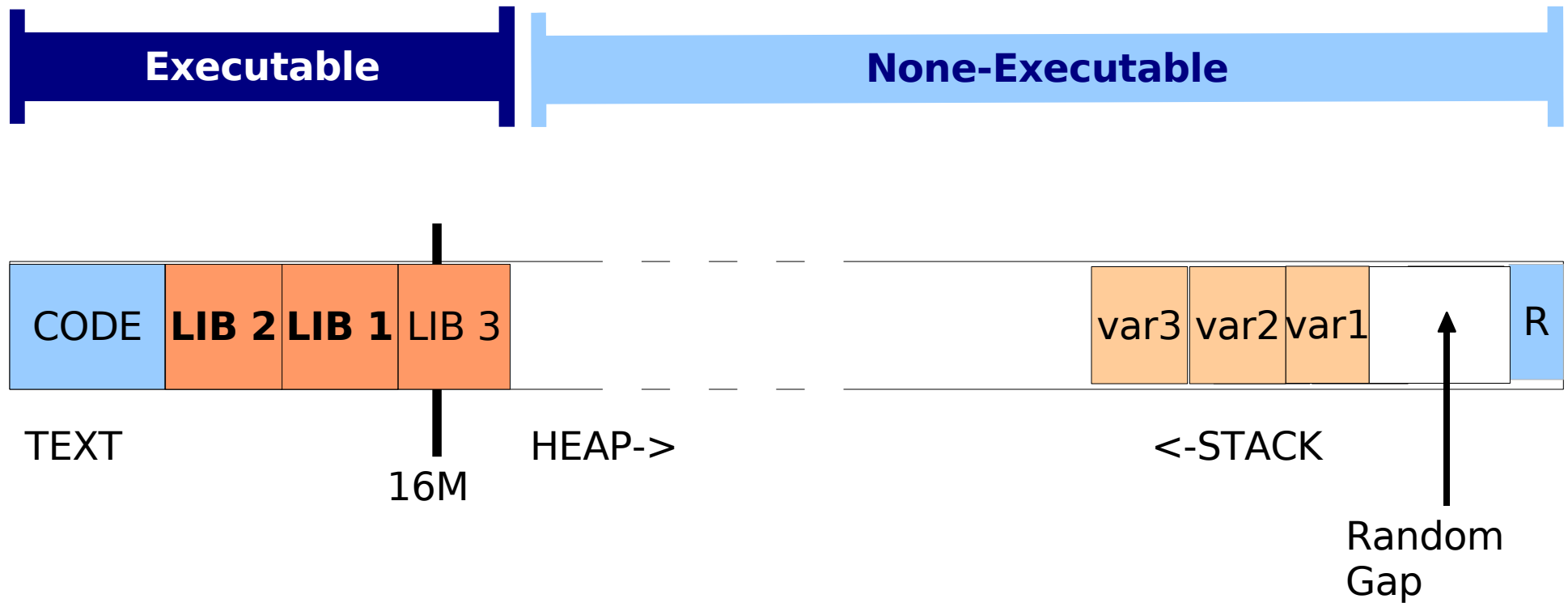


Exec-Shield

- Resourcefriendly Kernel-Patch
 - Some parts in Vanilla 2.6 Kernel
 - First Implementation: Fedora Core 2
- Three Defense Strategies
 - Limitation of executable memory (Segment Limits, NX/XD)
 - Random Adresses für Stack,Heap and Libraries
 - Special Mapping for executable Memory



How does it work?





Limitations

- Minor Performance-Penalty
- Not a perfect defense
- Some programs need an executable stack
 - ELF-Header PT_GNU_STACK

```
# execstack -q /usr/sbin/httpd  
- /usr/sbin/httpd
```


Configuration of Exec-Shield



- Stack-Protection:
 - For marked programs only:
`# sysctl -w kernel.exec-shield=1`
 - For all programs (beware!):
`# sysctl -w kernel.exec-shield=2`
- Randomization:
`# sysctl -w kernel.randomize_va_space=1`



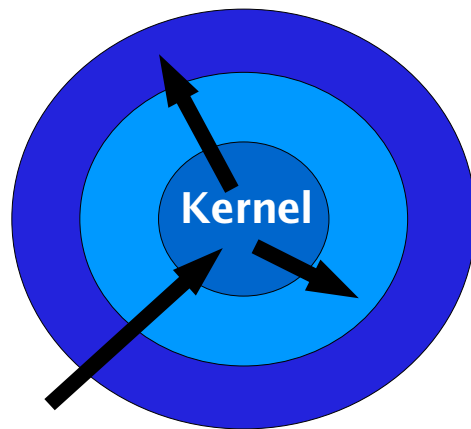
Additional Defenses

- Position Independent Executables (PIE)
 - Programs compiled as shared libraries
 - Random addresses for the binary
- ELF Data Hardening
 - Protects program headers
 - Arrays always stored after other variables
- Compiler and GLIBC Features
 - FORTIFY_SOURCE
 - Format String Exploits

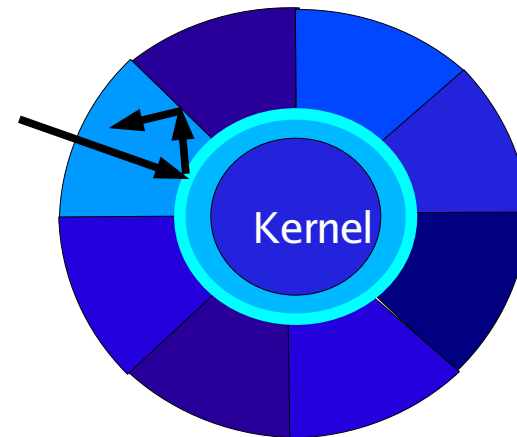


SELinux: Basics

- flexible Mandatory Access Control System
- Systemwide policy prevents access outside of a defined domain



Discretionary Access Control
Root exploit affects entire system



Mandatory Access Control
Kernel Policies define privileges of all processes
Thus an intruder cannot obtain full access even with root privileges



SELinux Basics II

- Building blocks
 - Subjects (processes)
 - Objects (files, system calls,..) marked by Security Contexts
- Commands
 - ls -Z, ps -Z
 - chcon
 - restorecon



Policy

- Enforced by the kernel
- Logging of all violations
- Uses White-Lists
- *targeted vs. strict*



Example: web server

- Apache runs under the domain of httpd_t
- Different File-Contexts:
 - httpd_sys_content_t: documents
 - httpd_sys_script_t: CGI Scripts
 - httpd_sys_script_rw_t: CGI generated files



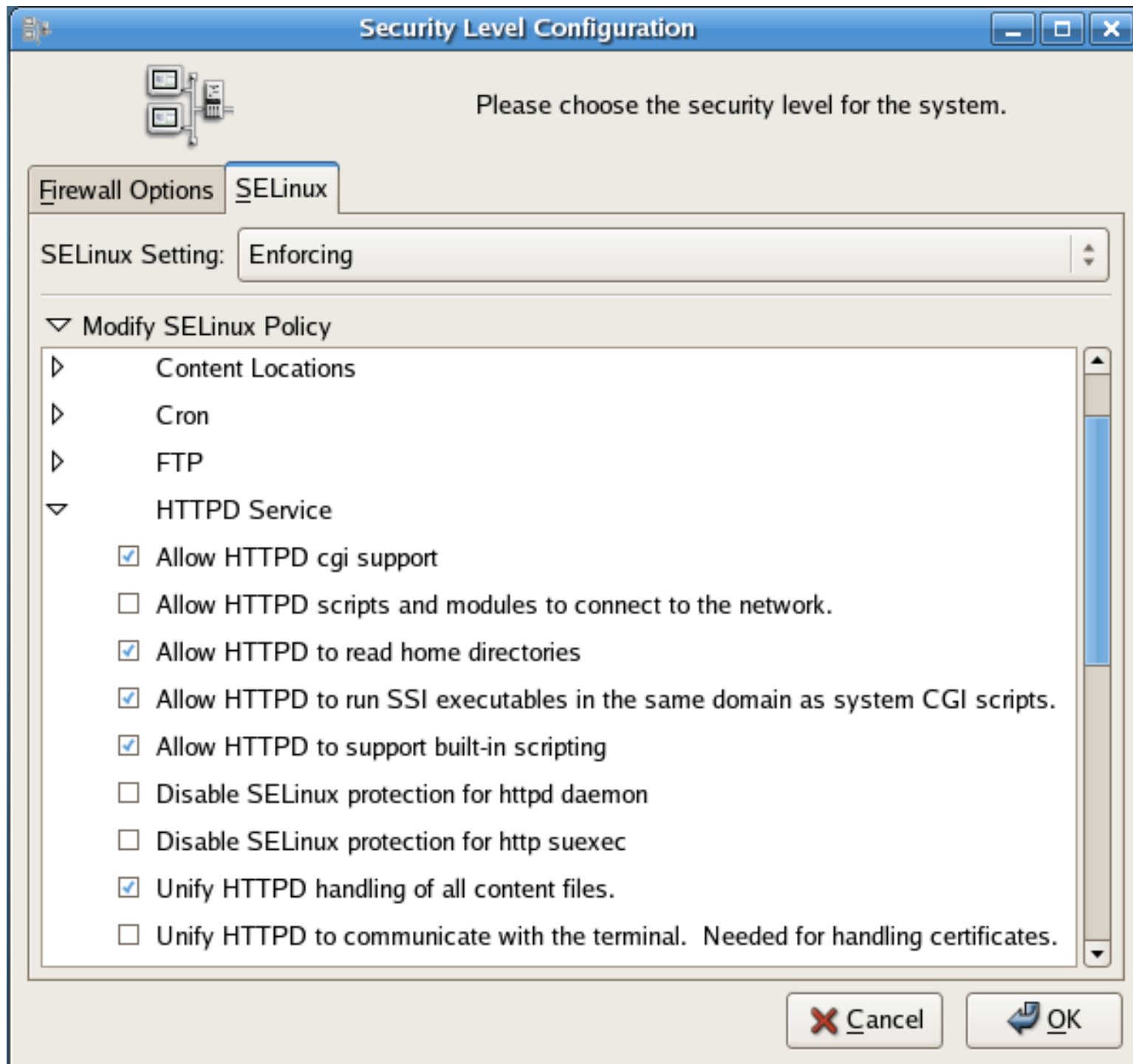
Rules of the Policy

- Only specific actions are allowed but not:
 - Execution of files generated by CGI:
`kernel: audit(1125333510.244:0): avc: denied {execute_no_trans} for pid=7660 comm=sh path=/tmp/foo dev=dm-2 ino=18
scontext=root:system_r:httpd_sys_script_t
tcontext=root:object_r:httpd_sys_script_rw_t tclass=file`
 - Opening of IP sockets by CGI scripts
`kernel: audit(1125463739.755:0): avc: denied { create } for pid=24071 comm=perl scountext=root:system_r: httpd_sys_script_t
tcontext=root:system_r:httpd_sys_script_t tclass=rawip_socket`
 - Reading log files or other critical data



Configuration of SELinux

- Policy violations are visible under `/var/log/messages`
- Certain policy parts can be toggled:
 - `setsebool`
 - `system-config-securitylevel`
- Setting of File contexts
`#chcon -R -t http_sys_content_t /webdata`
- Policy does not have to be changed under normal circumstances





Modular Policy

- parts of the policy can be changed independently
- Packages can install their own policy module
- Manage with: `semodule`

```
# semodule -l  
amavis    1.1.0  
ccs       1.0.0  
clamav    1.1.0  
dcc       1.1.0
```




Creating new Policy modules

- Create a temporary directory
- `touch <foo>.{te,if,fc}`
- Put your rules into `<foo>.te`:

```
policy_module(foo, 1.0)
require {
    attribute httpdcontent;
    type smbd_t;
}
allow smbd_t httpdcontent:dir create_dir_perms;
allow smbd_t httpdcontent:{ file lnk_file } \
    create_file_perms;
```



Compiling a new module

- Requires `selinux-policy-devel` and `checkpolicy`
- Compile the module with:
`make -f /usr/share/selinux/devel/Makefile`
- Install the module with:
`semodule -i <foo>.pp`



Example: audit2allow

- Without customization:
`# audit2allow -i /var/log/message -M local`
- With customiztation:
`# audit2allow -i /var/log/messages \
-m local > local.te`



Managing the Policy

- Semange modifies aspects of the policy
 - User mappings
 - Port/Interface mappings
 - File contexts
- Example: Grant access to a port to Apache

```
# semanage port -a -t httpd_port_t -p tcp 8000
```



Multi Level Security

- Provides sensitivity levels
- Categories define need-to-know basis
- Designed for governmental/military use
- Requires a new policy to be installed
- Restricts all process
- Can only alter enforcement, policy during boot
- Disables X, XFS



Multi Category Security

- Part of the standard policy
- Basically a MLS with a fixed sensitivity
- Privilege vs. Clearance
- All files are in the same category by default
- mcstransd daemon translates numeric values



Usage of MCS

- Creating categories:
semanage translation -a -T FooCat s0:cX
- Assign users to a category with:
semanage login -a -r FooCat username
- Assign files to a category
chcat +FooCat file



Dual Powers

- Issue: root can reassign categories
- Solution: Separate “SecAdmin”
 - Create a SecAdmin category
 - Grant users SecAdmin clearance
 - Allow users sudo access to semanage
 - Protect semanage with:
chcat +SecAdmin semange
 - Possibly protect chcat, chcon

Thanks for your attention !

